# Congestion avoiding mechanism based on inter-domain hierarchy

Weisser Marc-Antoine[1], Tomasik Joanna[1], and Barth Dominique[2]

[1] Supélec, 3, rue Joliot-Curie, F-91192 Gif-Sur-Yvette, France
{marc-antoine.weisser, joanna.tomasik}@supelec.fr
[2] PRiSM, University of Versailles,
45, avenue des États Unis, F-78035 Versailles, France
dominique.barth@prism.uvsq.fr

**Abstract.** Inter-domain routing is ensured by BGP. BGP messages carry no information concerning quality parameters of routes. Our goal is to provide domains with information regarding the congestion state of other domains without any changes in BGP. A domain, which is aware of congested domains, can choose a bypass instead of a route exhibiting possible QoS problems. We propose a distributed mechanism sending alert messages in order to notify domains about other domains congestion state. Our solution avoids flooding the Internet with signaling messages. It limits the number of alerts by taking advantage of the hierarchical structure of the Internet set by *P2C* and *P2P* relationships. Our algorithm is heuristic because it is a solution to an NP-complete and inapproximable problem. We prove these properties using the Steiner problem in directed acyclic graphs. The simulation runs show that our mechanism significantly diminishes the number of unavailable domains and routes compared to those obtained with "pure" BGP routing and with a theoretical centralised mechanism.
**Key words:** inter-domain, QoS, distributed algorithm

## 1 Introduction

We consider two paradigms for the introduction of QoS into the Internet: flow-based and connectionless. The first one is based on an individual flow management. Generally speaking, the flow-based paradigm is not adapted to the inter-domain level. This paradigm may be applied to a small number of critical requests which need strict QoS requirements. For the other demands of QoS without strict guarantee, the second paradigm which is connectionless, should be used. The principle of the connectionless paradigm is represented by the rejection of flow management. Packets are divided into classes and the *priority traffic*, packets of the most privileged class, is treated with priority. In [1] we proposed an algorithm founded on the flow-based paradigm which finds multi-constraint paths in an inter-domain network. In this paper we focus on the second approach.
**Our proposition:** BGP is a protocol which constructs routing tables in a inter-domain network. We take as a hypothesis that a domain using BGP arbitrarily

chooses one path from a set of possible paths i.e., a domain uses the same BGP routing table for all its border routers. If a domain was aware of the congestion state of the other domains, it could choose a path avoiding the congested domains. Yet, such an approach is unrealistic because 1) it would generate too many control messages and 2) domain operators would not want to give details of configuration and resources of their domains. We propose a mechanism which is external to BGP and which allows us to select paths avoiding congested domains. Our mechanism uses incomplete congestion state information to find bypasses which go around congested domains. Our mechanism copes with the two problems stated above. It could be implemented in a path computation element (PCE) [2] since PCE has to have a global vision of inter-domain paths.

The studies [3–6] of BGP tables show a hierarchical structure of the inter-domain network. This hierarchy is induced by the types of relationships connecting two domains: *P2C* (*provider to customer*) and *P2P* (*peer to peer*). In our mechanism, we use this hierarchy to limit the number of control messages.

A domain using our mechanism needs to know congested domains which are found only in its neighborhood. Moreover, a commercial contract behind relationships such as *P2C*, legitimates the exchange of the information concerning congestion. Providers payed by their customers should warn them about congestion in order to allow them to change their routing and use networks of other providers. The second problem stated above is solved because only a minimum of information is sent to a small number of domains. In this paper we only focus on the simulation results of our solution. The problem complexity results are presented in [7].

**Related works:** An investigated approach to introduce QoS into the inter-domain consists in either exhanging QoS characteristics between domains [8] or guessing the QoS characteristics using probe messages [9]. These approaches need a large amount of information difficult to collect and to use. In our approach we propose to use a binary information whether a domain is congested or not. It is exchanged only by connected nodes.

An utilization of hierarchy in QoS issues appears in some works [10,11]. The hierarchy described in [10] is artificial and it has be contructed. In our proposition, we use the existing hierarchy introduced by *P2P* and *P2C* relationships. We find a paper exploring this hierarchy in the routing context [11]. Its authors use it for a hybrid routing mechanism which works both with link-state and path vectors. To the best of our knowledge, the proposed utilization of alert message diffusion conditioned by the inter-domain hierarchy has never been studied.

**Paper organization:** We give a description of an inter-domain network and its hierarchy in the next section. We model our problem with the graph theory in Section 3 and give the results about its complexity and inapproximability. In Section 4 we describe our mechanism. In Sections 5 and 6 we present the simulations and their results. In order to judge our algorithm, we confront it with two extreme solutions: BGP and an on-line theoretical centralized algorithm which we define in order to compare purposes. Finally, we make conclusions and outline perspective future works.

## 2 Problem modeling

The problem modelling takes advantage of a specific structure of the inter-domain network which is composed of independent domains administrated by operators. Links connecting domains are characterized by two types of relationships: *P2C* and *P2P*. A *P2C* relationship links providers which sell connectivity to their customers. A *P2P* relationship exists between two domains which share connectivity. The studies of BGP tables [3–5] show that these types of relationships introduce a hierarchy in the inter-domain network.

There are layers which compose the hierarchy. The core, *Tier-1*, on the top of the hierarchy, is a set of domains which are linked together by *P2P* relationships. The domains of the core are not customers of any domain. Usually, the domains in the layer *Tier-*$j$ are providers of domains in the layer *Tier-*$k$ and customers of domains in the *Tier-*$i$, $i < j < k$. The domains in the layer *Tier-*$i$ and in *Tier-*$j$ are linked together with *P2C* relationships. Domains in the same layer can be linked together with *P2P* or *P2C* relationships.

The inter-domain hierarchy has an impact on the current inter-domain routing because routes are established according to commercial relationships. The only routes present in an inter-domain network are composed of an uphill and a downhill component. A downhill component is a list of consecutive links labelled with a *P2C* relationship. An uphill component is a list of consecutive links labelled with a *C2P* relationship which is dual to a *P2C*. The uphill and downhill components are connected either by zero or by one *P2P* relationship. Such routes are called *valley-free* by Gao [5]. We apply this term later in the paper.

We use a graph to model our problem. Let $G = (V, A, E)$ be a mixed graph (partially oriented [12]) to which we refer later in this paper as *inter-domain graph*. A vertex $v \in V$, represents a domain. We enumerate the elements of $V$ with natural numbers $1, \ldots, n$. An ordered pair $(p, c) \in A$ represents an existing *P2C* relationship between a provider $p$ and its customer $c$. A pair $\{p_1, p_2\} \in E$ represents a *P2P* relationship between domains $p_1$ and $p_2$.

A provider $p$, which sells a connectivity to a customer $c$, cannot be a customer of $c$. We, therefore, consider that there is no cycle made of *P2C* relationships in an inter-domain network [13]. This means that $(V, A)$ is a directed acyclic graph. The roots of this DAG are the vertices representing the domains of the core.

We define a capacity function $c : V \to \mathbb{R}$. The value $c(v_i)$ represents an amount of traffic which can be transited by $v_i$ without overloading it. Let $T : V^2 \to \mathbb{R}$ be a traffic matrix. Each value $T_{i,j}$ represents an amount of traffic which has to be sent from $i$ to $j$. The traffic $T_{i,i}$ represents the internal traffic of the domain $i$. Let $R : V^2 \to \mathbb{N}$ be a routing matrix. Each value $R_{i,j}$ represents the next hop for a packet passing through the domain $i$ which is to be sent to the domain $j$. We set $R_{i,i} = i$ for internal traffic. We say that a matrix $R$ is *valley-free* if, and only if, all the routes which it represents are valley-free.

Given a network, a traffic matrix, and a routing matrix, we say that a node is *perturbed* if the total amount of traffic transiting through it, emitted by it, and sending to it is greater than its capacity. A *perturbed path* is a path containing at least one perturbed node. Each $T_{i,j}$ is transmitted on a route induced by the

routing. Traffic from $i$ to $j$ is perturbed if the route from $i$ to $j$ is perturbed. The *volume of a perturbed traffic* is the sum of traffic passing on perturbed paths.

Given a network and a traffic matrix, our problem is to find a valley-free routing matrix which minimizes the number of perturbed nodes (*network approach*) and the volume of the traffic passing along perturbed paths (*traffic approach*).

This is an optimization problem with bi-criteria. We focus on the network approach only because we assume that minimizing the number of perturbed nodes may be a good heuristic approach to minimize the number of perturbed paths as well as the volume of perturbed traffic (see Section 5). Another reason for choosing the network approach criterion is implied by the operation mode of the proposed algorithm. According to the algorithm, nodes modifying their state send alert messages (see Section 3 for details). The minimization of the perturbed node number limits the number of sent messages.

Given a network and a traffic matrix, we refer to the above described problem as the *valley-free routing problem*. The weight of its solution is the number of perturbed nodes. In [7] we prove that this problem is *NP*-complete and inapproximable. We use the *directed Steiner tree problem* for the proofs.

## 3 Alert sending algorithm

Our distributed algorithm is based upon the principle of sending alert messages which carry information about the domain congestion state. To avoid taking the risk of flooding the entire network with alert messages, we aim to limit the range of their diffusion. We propose to take advantage of the inter-domain hierarchy to restrict the alert diffusion range. Each node informs its customers and providers when it becomes perturbed in order to allow them to change their routing. Each node also keeps them informed when it returns to an operational state.

Each node is provided with a BGP table and a *priority table* which stores potentially congestion-free routes set aside for priority traffic. The priority table is same as the BGP table if any node is not perturbed and its providers as well. The BGP table is altered by classical BGP mechanisms only [14]. The priority table is altered by our alerts and, occasionally, by changes in BGP table.

Each node contains a list of possible next hops towards every destination. These lists are constructed with routes announced by BGP. Their construction guarantees that all next hops satisfy the valley-free property of routes.

Each node can be in two distinct states: *green* or *red*. A node state is *red* if at least one of the two following conditions is satisfied: the amount of traffic transiting through the node, sending from it and sending to it, is greater than its capacity or at least one of its next hops which is a provider is in *red* state.

A node cannot become *red* because of its customer and peer congestion. Thanks to this fact, our algorithm avoids spreading of *red* nodes in all the network when a single node become *red*. We use the hierarchy to limit the number of nodes in *red* state as well as the messages sent: a node which has a customer or a peer in *red* state as a next hop should not be in *red* state itself. The node which is not in *red* state is in *green* state. The *green* state is a domain stable state.

Each node stores a table containing states of its neighbors. This table is updated by alerts sent by the neighboring providers and customers when their state changes. A node changing its state sends messages to its customers to inform them about its new state. The alerts are sent up and down in the hierarchy, but the peers do not receive the message so that the spread of messages would remain limited. The message is composed of: an ID of the node (domain number), a new state, and a delay $d$. It enters into the scheduler of the node receiving the message where it is processed after the delay $d$. The received message replaces any older message which arrived from the same node and is present in the scheduler.

We introduce a notation in order to describe the behavior of a node processing an alert message. Let $BR_i$ and $PR_i$ be tables stored in the node $i$, containing the next hop for the BGP and the priority routing, respectively. The values contained in $BR_i$ are computed by the BGP classical mechanism. The default values of $PR_i$ are $BR_i$. Let $st_i$ be a table stored in the node $i$ containing the known states of its neighbors. The default values in this table are *green*.

Let $LR_i$ be a table containing lists of the next hops which may be used to reach every destination. These lists are constructed using the routes announced by BGP. For a destination $j$, if there is a path from $i$ to $j$ via a customer of $i$, $LR_i[j]$ contains all next hops which are customers and which are announced by BGP. If a next hop is a peer, $LR_i[j]$ contains all the next hops announced by BGP which are peers. Otherwise, $LR_i[j]$ contains all next hops which are providers. These restrictions are useful to keep valley-free property preserved.

Algorithm 1 details the behavior of a node $i$ treating a message $m$ received from the node $j$. Steps 3–13 are executed when an alert arrives from a node $j$ which is in *red* state. The node $i$ selects then destinations for which the node $j$ is a designated next hop. The node $i$ then looks for alternative next hops for the selected destinations. An alternative next hop is chosen uniformly among domains stored in the set $LR_i[dest]$ and indicated as being in *green* state. If no node can be chosen, we use the default path stored in the BGP table in spite of its state. A valley-free route does not include any cycle. Our mechanism keeps this property preserved because the construction of the $LR_i$ table permits us to replace a customer only by a customer and a provider only by a provider.

Steps 14–24 are executed when the node $j$ is in *green* state. In this situation, the node $i$ replaces the nodes mentioned below by the node $j$: 1) the next hops whose state is *red* and which are leading to destinations for which $j$ can be a next hop, or 2) the next hops whose state is *green* towards all destinations for which $j$ is a "natural" next hop according to BGP routing.

In dynamical systems such as networks, oscillations can emerge. In our approach, this objectionable phenomenon may be introduced by exchanges of *green* and *red* alert messages. We avoid such instability by introducing a delay before the alert message is processed. Each alert message contains a field to store the delay before the message processing. The messages are sent immediately but the processing is deferred. The delay is set according to a random distribution in order to avoid synchronization in the network. We use an exponential distribution. The mean of the distribution is small for *red* alert and big for *green* alert.

**Algorithm 1** Node $i$ treating a message $m$

**Input:** message $m = (j, color, delay)$; tables $BR_I$, $PR_i$, $LR_i$, $st_i$
**Output:**

```
1: delete m from the scheduler
2: st_i[j] ⇐ color
3: if color = red then
4:    for all dest ∈ V do
5:       if PR_i(dest) = j then
6:          let S = {x ∈ LR_i[dest]/st_i[x] = green}
7:          if  S = ∅ then
8:             PR_i(dest) ⇐ BR_i[dest]
9:          else
10:            PR_i ⇐ choose_uniformly_in(S)
11:         end if
12:      end if
13:   end for
14: else
15:   for all dest ∈ V do
16:      if st_i[PR_i[dest]] = red then
17:         if j ∈ LR_i[dest] then PR_i[dest] ⇐ j
18:      end if
19:      if BR_i[dest] = j then PR_i[dest] ⇐ j
20:   end for
21: end if
```

The delay mechanism protecting the network against instability may unnecessarily introduce the *red* state in too many nodes. Therefore, the BGP table is used by default when no *green* next hop can be found and instability in the network leads our mechanism to work temporarily as BGP.

Let us observe that given an instance of our problem: a network and a trafic matrix, the goal of a distributed algorithm is to find a stabilized routing matrix minimizing the number of perturbed nodes. The problem of finding a routing matrix in a centralized way is *NP*-complete [7]. Then, a distributed algorithm finding a stabilized routing matrix will need an exponential number of messages.

## 4 Simulations

We use simulation to evaluate the performance of our algorithm. Its performance will be compared to the performance of BGP and a theoretical centralized algorithm. The performance measures used to make comparisons are the numbers of perturbed nodes and paths, and the amount of perturbed traffic. We expect that BGP will provide the worst results and we will use the results to obtain an upper bound for the number of perturbed elements. We point out that a centralized algorithm, studied here for comparison purposes, cannot be considered as implementable in a network because of its complexity and the huge amount

**Algorithm 2** Centralized algorithm

**Input:** Inter-domain graph $G$ ; list $L$ of demands (source, destination, traffic)
**Output:**
1: **let** $R$ be a routing matrix containing no path
2: sort demands $L$ in decreasing order of traffic
3: **for all** $q_i = (s_i, d_i, t_i) \in L$ **do**
4: **let** $p_i$ = find preferred path for $q_i$
5: **if** satisfying $q_i$ using $p_i$ does not perturb node **then**
6: add $p_i$ to matrix $R$ ; reduce capacity of node in $p_i$ by $t_i$ ; remove $q_i$ of $L$
7: **end if**
8: **end for**
9: **while** $L \neq \emptyset$ **do**
10: **let** $(s_j, d_j, t_j)$ = find the worst demand in $L$
11: $p_j$ = find best path for $(s_j, d_j, t_j)$
12: add $p_j$ to matrix $R$ ; reduce capacity of node in $p_j$ by $t_j$ ; remove $(s_j, d_j, t_j)$ of $L$
13: **end while**
14: return the routing matrix $R$

of data it has to process. We use it only to find a lower bound for the number of perturbed network elements.

**Centralized algorithm:** Given an inter-domain graph $G$ and a traffic matrix $T$, our goal is find a routing matrix which minimizes the number of perturbed nodes in a network. Thus, this centralized algorithm does not consider the traffic approach i.e., it does not take under consideration the overload of nodes. This problem is *NP*-complete and inapproximable. We cannot use any exact algorithm to solve it even for small instances. Thus, we have to use an heuristic algorithm.

We observe that paths rising in the hierarchy are longer than the other paths. Moreover, they pass through nodes which may be used to satisfy many demands. These paths may cause the introduction of many perturbed nodes which degrade the network performance. For a traffic *demand* $(s, d, t)$, where $t = T_{s,d}$, we define that a *preferred path* is a path which: 1) minimizes the number of perturbed nodes; 2) is preferably composed of nodes from lower layers.

To find such paths, we use an exhaustive route exploration. Exhaustive exploration runs in exponential time but we are not interesting in the complexity of this algorithm. For most of the instances the complexity is polynomial. In practice, exploration time is quite short.

Our centralized algorithm is detailed in the pseudocode 2. The steps 1–10 are the first phase of the algorithm during which we sort the demands in decreasing order of traffic amount and we satisfy as many demands as we can without perturbing any node. The introduction of any demand which is not yet satisfied (steps 11–18), produces perturbed nodes. Until there are still unsatisfied demands, we search the worst demand and we satisfy it using a preferred path. We define the *worst demand* as a demand: 1) whose satisfaction using the perfered path introduces the maximum number of new perturbed nodes; 2) whose amount of traffic is maximal.

**Simulation plan:** Our algorithm was tested for inter-domain topologies randomly generated with SHIIP [15]. It introduces the domain hierarchy into flat inter-domain topologies generated by BRITE [16]. The chosen topologies have parameters (core size, layer size, node degree) close to the means of a series of topologies of a given size obtained with SHIIP. The results discussed in the next section were obtained for a network of 100 domains.

A traffic matrix $T : V^2 \rightarrow \mathbb{R}$ is initially empty. The matrix filling procedure is iterative. After $|V|^2$ iterations we consider the matrix as a full matrix. Starting from this point we vary the network load. We choose an $(i, j)$ pair where both $i$ and $j$ are independently and uniformly distributed among the elements of $V$. We then set a new traffic value $T_{i,j}$ using the same normal distribution as before ($T_{i,i}$ indicates a traffic to be carried inside the domain $i$). In our simulation we use $\frac{\alpha}{|V|^2}\mathcal{N}(\sqrt{5}, 5)$ limited to nonnegative values. The scaling factor $\alpha, \alpha \geq 1$ allows us to study series of experiments with a growing traffic load. We do not specify a traffic unit, we speak of *Traffic Unit* (TU).

The choice of the time scale is not essential because we are interested in differences between the performances of our algorithm and the performance two algorithms of reference is expressed in terms of number of perturbed nodes, paths, and amount of perturbed traffic. We choose of the exponential distribution with a mean equal to one time unit.

We also choose the capacities in generated topologies to study networks which are not over-dimensioned. On the Internet, the largest domains with the largest capacities are at the top of the hierarchy. We suppose that the domains in the core are never perturbed. Thus, we choose to set the infinite capacity for all of them. We arbitrarily fix the capacity for domains in the *Tier-2*, *Tier-3*, and *Tier-4* to 35, 28 and 14 TU, respectively.

The first performance measure is the number of perturbed nodes because it is also the optimisation criterion for our algorithm. The second one is the number of perturbed paths and the third one is the amount of perturbed traffic. Both are strongly influenced by the first one. We are also interested in particular performance measures inherent to our distributed algorithm. We count the number of messages sent and the number of network state changes where a network state is a vector containing current congestion states of all network nodes.

## 5 Results

In the first place we compare the number of perturbed nodes obtained for one simulation run with our distributed algorithm and BGP (Fig. 1). The volume of transiting traffic is heavy enough ($\alpha = 2.0$) to saturate about 10 percent of network nodes with BGP approach. Observe that the number of the perturbed nodes increases quickly and reaches eight nodes when traffic matrix is filled up. The number of perturbed nodes oscillates around this value. Our distributed algorithm diminishes about five times the number of perturbed nodes. The initial heavy perturbation of six nodes is quickly reduced. Notice also that the shapes of these two diagrams are very similar. The number of alerts sent depends not

only on the number of perturbed nodes but also on the layers in which these nodes are localized. The manifestation of four more perturbed nodes can cause the generation of the same number of alerts messages as the manifestation of one perturbed node but only if these nodes are situated far from the core.

Two other performance measures, the number of perturbed paths and the quantity of perturbed traffic, taken for the same network during the same simulation run, are depicted in Fig. 2. The results for our algorithm are significantly better than for BGP although our method does not optimize them directly.

To verify the functioning of our method and that of BGP we performed a series of simulation runs for the same network and traffic load. The averages of the numbers of perturbed paths and nodes, presented in Fig. 3, exhibit that our algorithm works on average five times better than BGP for the given network.

The influence of the traffic load on the performance of our algorithm and BGP is presented in Fig. 4. We used averaged series of simulation runs. Each series had a different load which varied traffic light load ($\alpha = 1.5$) to saturating load ($\alpha = 2.5$). We observe that saturating traffic $\alpha = 2.5$ is critical for the network because almost half of the network nodes are perturbed. In this case the performance of our algorithm approaches that of BGP.

The additional cost of our algorithm in relation to BGP is expressed in the number of transmitted alert messages. Their values, for one simulation and for varying traffic loads are plotted in an accumulative manner in Fig. 5. For any traffic load the number messages sent increases rapidly during the traffic matrix filling up period. Next, the number of sent messages increases at a much slower rate. The increase in the number of sent messages is more significant for heavier traffic loads because more nodes become perturbed. As message transmission causes a change of node states, the number of network state changes follows the same pattern as those of alert messages.

The methodology applied to the comparison of our distributed algorithm and BGP cannot be used for a confrontation of our distributed and centralized algorithms. Firstly, computations performed by the theoretical centralised algorithm for each change in the traffic matrix is unacceptably time-consuming. Secondly, the variation of values of the elements of the traffic matrix one by one (Subsection VI.B) could be seen as too advantageous for our distributed algorithm.

To evaluate our distributed algorithm, while retaining BGP as a practical reference, we start from an instant of the problem. It is selected as a typical network state together with the traffic matrix from the simulation runs which have been discussed before. Starting from this point the traffic matrix is fixed and we activate our distributed mechanism with routing matrices produced with BGP and our centralized algorithm. We observe the rate with which it diminishes the number of perturbed nodes. We compare these results with the number of perturbed nodes when we use BGP and the centralized algorithm alone.

We compute a lower bound which is the number of nodes perturbed by traffic whatever the routing matrix is. For any routing matrix, a node $i$ is perturbed if the sum of traffic sent by $i$ and sent to $i$ is greater than its capacity. We do not consider traffic transiting through it. This is a lower bound for our problem.

Fig. 6 presents results of our comparison averaged for series of initial instances with heavy traffic: $\alpha = 2.0$. We observe that the distributed algorithm optimizes routing matrices produced with both BGP and our centralized algorithm. The perturbation in the network is caused by the routing. A lower bound indicates that 0.6% of nodes only are perturbed whatever the routing is. This bound is reached by the combination of our centralized algorithm and the distributed one. This means that the bound is not only a lower bound but the optimum. With routing provided by BGP and our centralized algorithm, 9.6% and 4.3% of nodes are perturbed, respectively. This means that about $9.6 - 0.6 = 9.0$ and $4.3 - 0.6 = 3.7\%$ of nodes are perturbed only because of the transit traffic due to the BGP routing and the routing given by the centralized algorithm, respectively. The combination of BGP and our distributed algorithm is very efficient because the average percentage of perturbed nodes is stablized around one percent. This is a better result than that provided by the centralized algorithm alone and only twice greater than the optimum. This figure shows that the distributed algorithms reduces efficiently the number of nodes perturbed only by transit traffic. Moreover, this number is close to the optimal number when the initial routing matrix is provided by the centralized algorithm. The centralized algorithm provides solution which are good initial solution for the distributed algorithm but it is not an eligible solution in practice.

## 6 Summary and further studies

In this paper we have proposed a BGP independent distributed mechanism in inter-domain networks to find non-congested routes which provides new possibilities missing from the state-of-art inter-domain traffic engineering. We performed exhaustive simulation experiments to evaluate the efficiency of our distributed algorithm. The obtained results are encouraging and we are convinced that the proposed distributed algorithm is worth further study.

First, we would like to verify the performance of our distributed algorithm for an inter-domain network model with the size and the hierarchy of the Internet. An open issue concerning the Internet hierarchy is notably the presence of *sibling to sibling* relationship which we have not investigated yet.

Second, our model assumes that the entire domain can be only in one of two states. An alternative way is to consider that a domain could be partially congested. In this case, alert messages could be sent to a subset of client domains instead of all the client domains. This modification of our algorithms allows us to reduce the number of sent messages and routing table alterations.

Third, we started our work from the hypothesis that information concerning congestion in the network allows an operator to route priority traffic on congestion-free paths. Notice that this is not a unique possibility to take advantage of the results of our distributed algorithm. An operator might prefer to place priority traffic on the best routes at his disposal despite their congestion state. In such a case, the operator redirects non-priority traffic on non-congested paths while saving the capacity of the best routes for priviledged traffic. The operator

decision may depend on route lengths and we would like to estimate them. The simulation studies presented in this paper consider inter-domain networks with traffic entirely seen as a priority traffic.

Fourth, another issue which we see as a subject of further studies is the performance of our distributed algorithm in the case of its partial deployment in an inter-domain network. Our preliminary opinion is that it is naturally adapted to incremental deployment.

Finally, certain problems appear in our distributed algorithm. It would be interesting to see how to find lengths of delays before alert messages are processed depending on a given inter-domain network characteristic. We think that more careful choice of these values may reduce momentary network state flickering.

## References


1. Weisser, M., Tomasik, J.: A distributed algorithm for inter-domain resources provisioning. In: Proc. of EuroNGI. (Apr. 2006) 9–16
2. A. Farrel, J.-P. Vasseur, J.A.: RFC 4655 - A Path Computation Element (PCE)-Based Architecture. Technical report (Aug. 2006)
3. Ge, Z., Figueiredo, D.R., Jaiswal, S., Gao, L.: The hierarchical structure of the logical Internet graph. In: Proc. of SPIE ITCOM'01. (Jul. 2001) 208–222
4. Subramanian, L., Agarwal, S., Rexford, J., Katz, R.H.: Characterizing the Internet hierarchy from multiple vantage points. In: Proc. of IEEE INFOCOM'02. Volume 2. (Jun. 2002) 618–627
5. Gao, L.: On inferring autonomous system relationships in the Internet. IEEE/ACM Transactions on Networking **9**(6) (2001) 733–745
6. Battista, G.D., Erlebach, T., Hall, A., Patrignani, M., Pizzonia, M., Schank, T.: Computing the types of the relationships between autonomous systems. IEEE/ACM Transactions on Networking **15**(2) (Apr. 2007) 267–280
7. Weisser, M., Tomasik, J.: Innapproximation proofs for directed Steiner tree and network problems. Technical report, PRiSM UVSQ (Jun. 2007)
8. Cristallo, G., Jacquenet, C.: An Approach to Inter-domain Traffic Engineering. In: Proc. of WTC'02. (Sep. 2002)
9. Xiao, L., Lui, K., Wang, J., Nahrstedt, K.: QoS extension to BGP. In: Proc. of ICNP'02. (Nov. 2002) 100–109
10. Okumu, I., Mantar, H., Hwang, J., Chapin, S.: Inter-domain qos routing on diffserv networks: a region-based approach. Comp. Comm. **28**(2) (Feb 2005) 174–188
11. Subramanian, L., Caesar, M., Ee, C.T., Handley, M., Mao, M., Shenker, S., Stoica, I.: HLP: a next generation inter-domain routing protocol. In: Proc. of SIGCOMM'05, ACM Press (Aug. 2005) 13–24
12. Raghavachari, B., Veerasamy, J.: Approximation algorithms for mixed postman problem. SIAM J. on Discrete Mathematics **12**(4) (1999) 425–433
13. Gao, L., Griffin, T.G., Rexford, J.: Inherently safe backup routing with BGP. Proc. of INFOCOM'01 **1** (Apr. 2001) 547–556
14. Rekhter, Y., Li, T., Hares, S.: RFC 4271 - (BGP-4). Technical report (Jan. 2006)
15. Weisser, M., Tomasik, J.: Automatic induction of inter-domain hierarchy in randomly generated network topologies. In: Proc. of CNS'07. (2007) 77–84
16. Medina, A., Lakhina, A., Matta, I., Byers, J.: BRITE: An approach to universal topology generation. In: Proc. of MASCOT'01. (Aug. 2001) 346–354

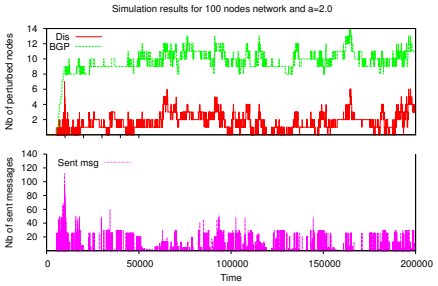


**Fig. 1.** The number of perturbed nodes for our algorithm (one simulation run, $\alpha = 2.0$).

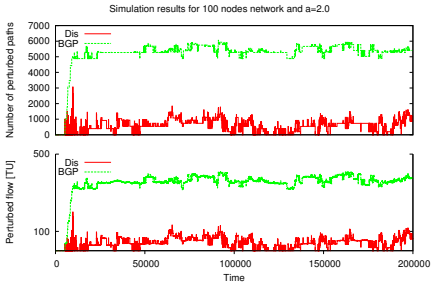


**Fig. 2.** Number of perturbed path and amount of perturbed traffic for our algorithm (one simulation run, $\alpha = 2.0$).

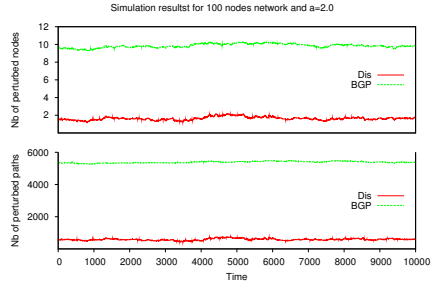


**Fig. 3.** The number of perturbed nodes and paths averaged for a simulation run series, $\alpha = 2.0$.

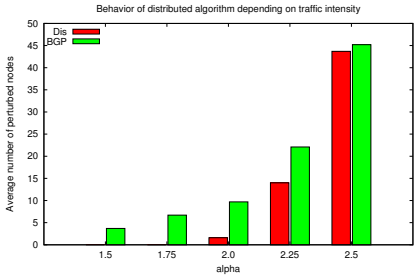


**Fig. 4.** The number of perturbed nodes depending on the traffic load averaged for series of simulation runs.

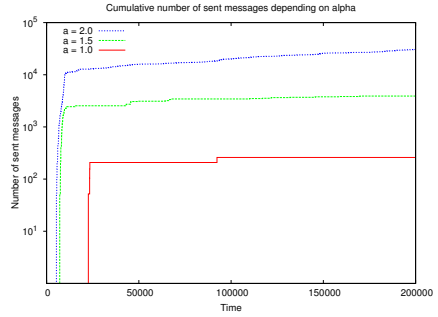


**Fig. 5.** The cumulative number of sent messages for one simulation run depending on the traffic load.

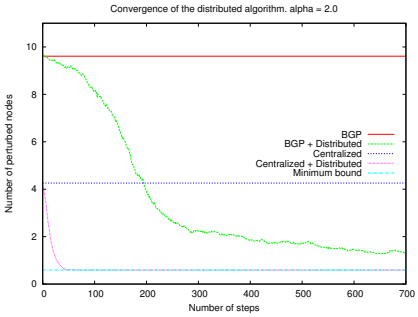


**Fig. 6.** The number of perturbed nodes averaged for series of initial instances, $\alpha = 2.0$.